Assignment 6

● Graded

Group

Kyle Hoop
Adam Fenjiro
View or edit group

Total Points

77 / 90 pts

Question 1

Q1

25 / 25 pts

✔ **– 0 pts** Correct

– 2 pts Movie PK wrong

– 1 pt Movies PK incorrect , should be (title, year)

– 2 pts MoviesExec PK incorrect or no PK

– 3 pts Missing foreign key on Movies

– 1.5 pts Missing on delete set null

– 1.5 pts Missing on update cascade

– 2 pts producerCNum has not null constraint

– 2 pts Foreign key on wrong table or wrong columns

– 2 pts Missing enum type for genre

– 1 pt Missing some options from enum

– 25 pts Wrong/Missing

– 5 pts Missing The MoviesExec table

– 2 pts Did not show enough evidence

– 1 pt some error, see marks with on the report

– 2 pts Did you run the query in your database? FK should be specified as reference_definition:
REFERENCES tbl_name (key_part,...)
[MATCH FULL | MATCH PARTIAL | MATCH SIMPLE]
[ON UPDATE reference_option]

Question 2

Q2 **28** / 40 pts

2.1 **2.1** **5** / 5 pts

✔ **– 0 pts** Correct

– 1 pt Missing the last_modified attribute in insertion

– 1 pt Missing a column in the table creation

– 3 pts Did not include SQL statements. From answers after this question, it looks like you did it. Could not check if you have used the correct data type, PK.

– 1 pt no insertion statement

– 1 pt no pk

– 5 pts missing

2.2 **2.2** **10** / 10 pts

✔ **– 0 pts** Correct

– 10 pts missing

– 1 pt no select statement

– 2 pts missing SQL update statements

– 1 pt Did not show evidence that the level is serializable

– 2 pts no begin tran

– 1.5 pts No rollback

– 1.5 pts no commit

– 6 pts Not a transaction

– 2 pts two updates should be in one transaction

– 1 pt select is not inside transaction

2.3 **2.3** **0** / 5 pts

– 0 pts Correct

✔ **– 5 pts** Missing

– 2 pts Not done as a explicit transaction

– 1 pt should be only two accounts , not all, not one. Should use two seperate select statements otherwise no need to use explicit transaction as each statement is a trasaction

– 2 pts two selection statements should be in one transaction

– 1 pt no commit

2.4 **2.4** **7** / 10 pts

- **– 0 pts** Correct
- **– 10 pts** Missing
- **– 5 pts** Did not get a timeout error or did not show it with screen copy
- **– 2 pts** Did not provide any explanation
- ✔ **– 3 pts** not a direct screen shot.

2.5 **2.5** **6** / 10 pts

- **– 0 pts** Correct
- **– 10 pts** Missing
- **– 0 pts** Did not provide an explanation
- **– 2 pts** Did not show the deadlock
- ✔ **– 4 pts** not screen shot of deadlock
- **– 8 pts** made some effort. but not right

Question 3

Q3 **9** / 10 pts

- **– 0 pts** Correct
- **– 10 pts** Missing
- **– 1.5 pts** 3.2 No SQL using the view
- **– 3 pts** 3.2 No answer
- ✔ **– 1 pt** missing @localhost account or @%
- **– 2 pts** 3.3 wrong syntax for username. should be 'user_name'@'host_name'
- **– 3 pts** 3.3 no answer
- **– 3 pts** 3.2 no answer or wrong answer

Question 4

Q4

15 / 15 pts

✔ **– 0 pts** Correct

– 2 pts Incorrect/missing part 1

– 1 pt Missing part 2

– 2 pts Part 3: Missing output before

– 2 pts Part 3: Missing/incorrect index mysql code

– 2 pts Part 3: Missing output after

– 1 pt Part 4.3.1: Missing/incorrect index mysql code

– 1 pt part 4.3.2 missing/incorrect index . Should be (item_id, sale_quantity)

– 2 pts Part 4: Missing out before

– 2 pts Part 4: Missing output after

– 1 pt 4.3.2 no before or after time

– 15 pts missing/empty

No questions assigned to the following page.

CS3425 Assignment 6

90 points

Goals:

1. Understand constraints - fk, pk, not null, unique,
2. Use more data types – data, time, enum, year
3. Use transaction and understand locks
4. Work with views
5. Create indexes
6. Use grant command

1. (25 points) Constraints and data types.

Create tables **Movies, MovieExec** in your own database by following the notes below. See 2.2.8 *An Example Database Schema* on page 26 for detailed explanation about each attribute.

```
0
Movies( title, year, length, genre, studioName, producerCNum)
MoviesExec(name, address, certNum, netWorth)
```

Notes:

a) Use **enum** for genre to include the common genre types such as “action”, “adventure”, “comedy”, “family”, “crime”, “drama”, “fantasy”, and “historical”. See https://dev.mysql.com/doc/refman/5.7/en/enum.html

b) Use **year** type for column year. See https://dev.mysql.com/doc/refman/5.7/en/year.html

c) Use **appropriate data types, primary keys, not null constraints** for all other columns.

d) The data referencing constraints:

- The producer of a movie must be someone mentioned in MovieExec.
- Modifications of certNum in MovieExec will cause the related producerCNum data in Movies to be changed accordingly.
- Removal of certNum in MovieExec will cause the related producerCNum data in Movies to be set as null.

Question assigned to the following page: 1

```
-- create Movies table
DROP TABLE IF EXISTS Movies;
CREATE TABLE Movies (
 title VARCHAR(25),
 year INT PRIMARY KEY,
 length TIME,
 genre ENUM('action', 'adventure', 'comedy', 'family', 'crime', 'drama', 'fantasy', 'historical'),
 studio_name varchar(20),
 producerCNum int REFERENCES MoviesExec(cert_num) ON UPDATE CASCADE ON DELETE
SET NULL
);

-- create MoviesExec table
DROP TABLE IF EXISTS MoviesExec;
CREATE TABLE MoviesExec(
name VARCHAR(25),
address VARCHAR(25),
cert_num int PRIMARY KEY,
net_worth int
);
```

2. (40 p) Transaction

2.1 (5 p) Create tables and insert data.

a) Create table account(acc_no, customer_id, balance, last_modified)

- Column acc_no is the primary key
- Column Last_modified has the timestamp data type
- Column balance has type of numeric(10,2).
-

b) Insert two accounts "A123" and "A345". For the column last_modifed, use the value of function now(). Like this:

insert into account values("A123","…", X , now());

2.2 (10 p) Implement the transaction of transferring some amount $X from A123 to A345 in SQL.

Implement the transaction in MySQL command line utility. No need to implement it in Java.

- Set the level to be serializable before you start the transaction
  See https://dev.mysql.com/doc/refman/8.0/en/set-transaction.html#set-transaction-isolation-level. Please set the session level. Otherwise the setting will only be valid for the next transaction (explicit or implicit transaction).

- Read the balance of A123 with select statement.

Questions assigned to the following page: 2.1 and 2.2

- "Check" manually (select the data, then visually verify) to see if the balance is less than $X. If true, rollback the transaction. Please note you will do the checking "manually" as we haven't learned how to code it yet. You may write pseudo code in your report or just comments to indicate the work. You will learn how to program it later.

- Update the balance for A123
- Update the balance for A345
- Commit the transaction.

You should include 2 cases in your report. Pick different $X such that:
Case1: $X is less than the balance in A123.
Case1: $X is more than the balance in A123.

2.1 a)

```
-- create account table
DROP TABLE IF EXISTS account;
CREATE TABLE account(
acc_no VARCHAR(5) PRIMARY KEY,
customer_id VARCHAR(10),
balance NUMERIC(10,2),
last_modified TIMESTAMP
);
```

2.1b)

```
-- insert into account table
INSERT INTO account VALUES('A123', 'Joe', 1500.25, NOW());
INSERT INTO account VALUES('A345', 'Sue', 2600.75, NOW());
```

2.2).0

```
-- create a serializable transaction
SET SESSION TRANSACTION ISOLATION LEVEL SERIALIZABLE;
START TRANSACTION;

-- check A123's balance manually
SELECT acc_no, balance
FROM account
WHERE acc_no = '';

-- create the transfer of the funds from A123 to A345 when X <= balence of A123
UPDATE account SET balance = balance - 100.00, last_modifed = now() where
acc_no = 'A123';

UPDATE account
SET balance = balance + 100.00, last_modifed = now()
where acc_no = 'A345';
COMMIT;

-- have a rollback for if the requested amount X > balence of A123
ROLLBACK;
COMMIT;
```

2.3

Questions assigned to the following page: 2.3 and 2.4

```
-- display the total balance of A123 and A345
select acc_no, balance
from account;
```

2.3 (5 p) Implement the transaction of displaying the total balance of A123 and A345

2.4 (10p) Lock Timeout

Design a case where two client transactions are accessing the database about the same time. One client transaction will be waiting for lock and get timeout error message. Experience the execution with actual data in Mysql until you get the timeout error message from the database server. Explain the case that you designed and show the screen copy in your report.

Note: Open two terminals and have one transaction runs in each terminal. Don't use two SQL tabs in MySQL WorkBench.

After you ssh or putty, connect to database with command below to get into the interactive mode.

mysql -hclassdb.it.mtu.edu -uyourusername -p yourdatabase

The first user is having a conversation while he opens the transaction:

```
SET SESSION TRANSACTION ISOLATION LEVEL SERIALIZABLE
+-----------------+
| @@transaction_isolation  |
+-----------------+
| REPEATABLE-READ |
+-----------------+

SELECT @@transation_ISOLATION
+-----------------+
| @@transaction_isolation  |
+-----------------+
| SERIALIZABLE |
+-----------------+

BEGIN;
UPDATE account SET balance = balance - 100 WHERE acc-no = 'A123'
UPDATE account SET balance = balance + 100 WHERE acc-no = 'A345'
```

The first user is still having a conversation, here is his view:

```
SELECT * FROM account;
+---------+-------------+---------+-----------------------+
| acc_no  | customer_id | balance | last_modified         |
+---------+-------------+---------+-----------------------+
| A123    | 1           | 1000    | 11-16-2023 22:53:04   |
| A345    | 2           | 1500    | 11-16-2023 22:53:46   |
+---------+-------------+---------+-----------------------+
```

Questions assigned to the following page: 2.4 and 2.5

At the same time, here is the second user's view:
SELECT acc_no, balance FROM account;

```
+---------+-------------+---------+-----------------------+
| acc_no  | customer_id | balance | last_modified         |
+---------+-------------+---------+-----------------------+
| A123    | 1           | 2000    | 11-16-2023 22:53:04   |
| A345    | 2           | 2500    | 11-16-2023 22:53:46   |
+---------+-------------+---------+-----------------------+
```

Here is a timeouy between the update and commit of the first user:
DO SLEEP(60);
COMMIT;

2.5 (10p)Deadlock.

Design a case where two client transactions will be waiting for locks that were hold by the other. Therefore, the system detects the waiting loop. One transaction gets deadlock error message and will be killed. Experience the execution with actual data in Mysql until you get the dead lock error message from the database server. Explain the case that you designed and show the screen copy in your report.

Note: Open two terminals and have one transaction runs in each terminal. Don't use two SQL tabs in MySQL WorkBench.

A company X has two database administrators. The manager asked both to update all customer_id+2. Both database administrators start the transaction at the same time.
Admin 1 updates A1234, then A345.
Admin 2 updates A345, then A123.
Since they are both performing the same task and admin 2 connected a few seconds after admin 1; admin 2 got the error:
UPDATE account SET customer_id = customer_id + 2 WHERE acc_no = 'A345';
UPDATE account SET customer_id = customer_id + 2 WHERE acc_no = 'A123';

ERROR 1214 (40001): Deadlock found when trying to get lock; try restarting transaction.

Question assigned to the following page: 3

3. (10 points) Views

3.1 Create a view **RichExec** to include all the movie executive's names whose netWorth is more than 1,000,000.

```
SELECT VIEW RichExec AS
SELECT name FROM MoviesExec
WHERE netWorth > 1000000;
```

3.2 Write a SQL to list all the rich executive names using the view **RichExec**

```
SELECT * FROM RichExec;
```

3.3 Use grant command to give our TA read access to view RichExec.

See https://dev.mysql.com/doc/refman/8.0/en/grant.html
Here are the TA's accounts:

```
mysql> select user,host from mysql.user where User like 'junyaoy' order by user\G
*************************** 1. row ***************************
user: junyaoy
host: %
*************************** 2. row ***************************
user: junyaoy
host: localhost
2 rows in set (0.08 sec)
```

```
GRANT SELECT ON RichExec to 'junyaoy'@'%';
```

Question assigned to the following page: 4

4. (15 points) Index

4.1 What is the use of an index?

A SQL index is used to retrieve data quickly from a database. Indexing a table/view is one of the most effective techniques to increase query and application performance. An SQL index is a quick lookup table that is used to find records that users commonly search for.

4.2 Copy the table sales from University to your own database. The table has 2.4M rows, so it will take a bit of time. See here for how to use "create table … select " https://dev.mysql.com/doc/refman/8.0/en/create-table-select.html

Tip: set the working database to be your own database, then use university.tablename to get data from table in university database.

```
CREATE TABLE university_sales SELECT * FROM university.sales;
```

4.3 Create indexes to speed up the following queries. Show the index and the execution time difference before and after the index.

4.3.1 Find the total number of sales that belongs to a particular customer;

```
select count(*) from sales  where customer_id=4506;
```

Before: count(*) = 29, execution time = 1.68sec

```
CREATE INDEX customerIndex ON university_sales(customer_id);
SELECT count(*) FROM university_sales
USE INDEX (customerIndex)
WHERE customer_id = 4506;
```

After: count(*) = 29, execution time =: 0sec

4.3.2 Find the most popular item_id and its sale quantify

```
select item_id, sum(sale_quantity) from sales group by item_id order
by sum(sale_quantity) desc limit 1 ;
```

Before: item_id = 2466, sum(sale_qunatity) = 3512, execution time = 1.85sec

```
CREATE INDEX itemsIndex ON university_sales(item_id, sale_quantity);
```

After: item_id = 2466, sum(sale_qunatity) = 3512, execution time = 10.72sec

4.4 Drop the table that you copied. Because we have many students in the class, too many copies of this table will cause the database space get filled up quickly.

```
DROP TABLE university_sales;
```